Generación Automática de Aplicaciones Mixtas Sw/Hw mediante la Integración de Componentes COTS

Cristina Vicente, Ana Toledo, Carlos Fernández, Pedro Sánchez

***Resumen*-- Los grandes avances realizados en el mundo de la electrónica han dado lugar a la proliferación de sistemas mixtos que combinan la flexibilidad de las rutinas Software (Sw) con la velocidad del procesamiento Hardware (Hw). En este artículo se presenta un nuevo enfoque para el desarrollo de este tipo de sistemas mixtos Sw/Hw, que cubre todas las fases de su ciclo de vida. La propuesta que aquí se recoge integra dos de los paradigmas de desarrollo de software más en boga actualmente: el desarrollo de software basado en componentes y la generación automática de software (meta-programación). Si bien este trabajo se centra en el desarrollo de aplicaciones mixtas Sw/Hw en el dominio de los Sistemas de Procesamiento de Información Visual (VIPS), los resultados obtenidos son fácilmente trasladables a otros dominios como los de las aplicaciones de control o de telecomunicación. Como parte de este trabajo se ha desarrollado la herramienta visual IP-CoDER que permite la construcción incremental de VIPS desde el diseño de prototipos funcionales y su particionado en módulos Sw/Hw, hasta la generación automática del código final de la aplicación.**

***Palabras clave*-- Generación Automática de Software, Desarrollo de Software Basado en Componentes, Componentes COTS: Commercial Off-The-Shelf, Sistemas Mixtos Sw/Hw, Sistemas de Procesamiento de Información Visual (VIPS)**.

Este trabajo ha sido parcialmente financiado por el Proyecto Europeo *EFTCOR* (DPI2002-11583-E), y por los proyectos CICYT *COSIVA* (TIC 2000-1765-C03-02) y *ANCLA* (TIC 2003-07804-C05-02).

Los autores son miembros del Grupo de Investigación DSIE de la Universidad Politécnica de Cartagena, Campus Muralla del Mar, Edificio Antigüones, Cartagena, España. Teléfono: +34968326448. Email: Cristina.Vicente@upct.es.

I. INTRODUCCIÓN

Actualmente, son muchos los dispositivos Hw que requieren de una lógica cada vez más compleja y sofisticada. Así, hoy en día encontramos en el mercado desde teléfonos móviles capaces de transmitir vídeo en tiempo real, hasta frigoríficos "inteligentes" capaces de realizar la compra de productos alimentarios a través de Internet. La programación de estos dispositivos suele realizarse utilizando lenguajes propietarios de muy bajo nivel, optimizados para obtener el máximo rendimiento de la plataforma Hw específica empleada en cada producto (microprocesadores, memorias, buses, Digital Signal Processors DSP, Field Programmable Gate Array FPGA, etc).

Los grandes avances realizados en el campo de la electrónica nos permiten contar en la actualidad con Hw cada vez más flexible, potente, pequeño y económico. Sin embargo, algunos sistemas requieren del uso de algoritmos complejos que deben procesar ingentes cantidades de información, a veces bajo estrictos requisitos de tiempo real. En este caso, la implementación exclusivamente Hw de este tipo de sistemas puede resultar desaconsejable, cuando no inviable, por varios motivos. En primer lugar, si bien en términos generales el procesamiento Hw es mucho más rápido que el equivalente Sw, no es sencillo codificar, menos aún de manera optimizada, ciertos algoritmos complejos haciendo uso de lenguajes de tan bajo nivel como los requeridos por este tipo de dispositivos. Es más, codificar y optimizar estos algoritmos puede requerir tanto tiempo que, al ritmo que evoluciona la electrónica, muy probablemente aparecerán en el mercado dispositivos mucho más potentes sobre los que dichas optimizaciones tendrían un impacto prácticamente despreciable. Además, a pesar del abaratamiento de estos dispositivos, la complejidad de ciertos sistemas requiere del uso

de ciertas tecnologías demasiado costosas aún como para resultar viables.

Por todo ello, en ocasiones resulta imprescindible abordar la construcción de este tipo de sistemas a partir de un diseño mixto Sw/Hw. De este modo, parte del procesamiento, generalmente el más complejo, se codificará haciendo uso de lenguajes de programación de alto nivel y se ejecutará sobre un procesador convencional (microprocesador). Simultáneamente, ciertos cálculos sencillos y que deban realizarse de manera muy veloz, se ejecutarán en paralelo sobre un dispositivo Hw de propósito específico (por ejemplo, un sensor inteligente) o más general (por ejemplo, una FPGA).

En particular, entre los sistemas de este tipo, nuestro grupo de investigación cuenta con una amplia experiencia en el desarrollo de Sistemas de Procesamiento de Información Visual (VIPS) [1-2]. En este artículo se analizará cómo el uso de técnicas de generación automática de código [3] y de desarrollo de Sw basado en componentes [4], en particular de tipo COTS [5], puede ayudar a diseñar e implementar este tipo de sistemas de manera rápida, sencilla e intuitiva. A pesar de que este trabajo se centra en el dominio específico de los VIPS, la metodología que aquí se presenta es de aplicación para una amplia variedad de sistemas mixtos Sw/Hw, como los sistemas de control en tiempo real, los sistemas de telecomunicación, o las redes de sensores, entre otros.

El resto del artículo se estructura como sigue. En el siguiente apartado se describe la metodología tradicionalmente empleada para el desarrollo de los VIPS y los inconvenientes que de ella se derivan. En el apartado 3 se propone un nuevo enfoque metodológico que permite realizar diseños menos dependientes del Hw y por lo tanto más flexibles y fáciles de reutilizar. Para dar soporte a esta nueva metodología se ha creado la herramienta visual IP-CoDER, que se presenta en el apartado 4. Esta herramienta permite diseñar e implementar nuevos VIPS de manera semiautomática, haciendo uso de una librería de componentes Sw y Hw para procesamiento de imágenes, construida a partir de diversos componentes COTS. Finalmente, en el apartado 5 se recogen algunas conclusiones y líneas de trabajo futuras.

II. SISTEMAS DE PROCESAMIENTO DE INFORMACIÓN VISUAL (VIPS)

En la actualidad, el abanico de aplicaciones que hacen uso de información de tipo visual es amplísimo: sistemas de inspección visual automatizada de productos industriales, aplicaciones de videoconferencia, sistemas de seguridad biométricos basados en la identificación del rostro o de las huellas digitales, sistemas de diagnóstico médico, aplicaciones de compresión de vídeo, etc.

El procesamiento de imágenes es una tarea computacionalmente muy costosa dado el ingente volumen de información contenido en cada imagen y la complejidad de algunos de los algoritmos necesarios para extraer la información relevante en cada caso. Por ello, la implementación de VIPS suele requerir del uso tanto de lenguajes de programación eficientes como de procesadores potentes. En la actualidad, el mercado ofrece diversos productos, tanto Sw como Hw, útiles para el desarrollo de VIPS:

- Librerías de procesamiento de imágenes [6-8] optimizadas para distintas plataformas:
  - Plataformas Sw de propósito general como los procesadores de Intel©, o de propósito específico como algunos DSP.
  - Plataformas Hw de propósito general como las FPGA, o costosas plataformas de propósito específico basadas en DSPs (p.e. los sistemas de Matrox© [7]).
- Herramientas de programación, ya sean generales (por ejemplo, cualquiera de los entornos de desarrollo C/C++) o específicas para el desarrollo de VIPS [9].
- Herramientas de (co-) simulación [10-11].

Sin embargo, ninguno de estos productos cubre por sí sólo todas las fases del desarrollo de los VIPS. Así, a fecha de hoy, este tipo de sistemas se siguen codificando de manera casi artesanal, siguiendo una metodología muy centrada en el Hw, tal y como se describe a continuación.

A. Construcción tradicional de los vips

Tradicionalmente, los VIPS se construyen en torno a un determinado Hw, a pesar de tratarse de sistemas intensivamente Sw. De hecho, la práctica habitual consiste en seleccionar, en primera instancia, la plataforma Hw sobre la que se implementará el VIPS, atendiendo a ciertos requisitos no funcionales como el coste máximo del sistema final o la velocidad mínima a la que deben procesarse las imágenes (ver Fig. 1), para posteriormente, desarrollar la lógica Sw atendiendo a la funcionalidad exigida al sistema.

La primera consecuencia negativa derivada del hecho de anteponer la selección del Hw a la implementación del Sw, suele ser la adquisición de plataformas sobredimensionadas, tanto en coste como en capacidad de procesamiento, haciendo válido el dicho “más vale que sobre”. Además, la elección y adquisición de una plataforma Hw como primer paso en la construcción de un VIPS, suele imponer el uso de determinados controladores (*drivers*) y librerías de procesamiento de imágenes optimizadas para dicha plataforma (ver Fig. 1). Sin embargo, quizá la limitación más importante resultado de la subordinación del Sw al Hw, sea la dificultad de conseguir diseños flexibles y reutilizables. Como consecuencia, cada nuevo VIPS se construye prácticamente desde cero, aunque su lógica sea casi idéntica a la de otros sistemas desarrollados previamente, pero implementados sobre otras plataformas. Una de las razones que podrían justificar la concepción Hw de los VIPS, es que este tipo de aplicaciones suelen implementarlas ingenieros en automatización y control con experiencia como programadores, pero con escasa formación en Ingeniería del Software. Así, en la mayoría de los casos, las fases de análisis y diseño o bien se obvian, o bien se realizan de forma poco sistemática sin seguir una metodología formal de desarrollo de Sw.

En cuanto a la fase de implementación, ésta suele realizarse en dos etapas. La primera de ellas consiste en desarrollar un prototipo Sw sobre el que testar distintos algoritmos de procesamiento de imágenes, seleccionados generalmente de manera empírica (prueba y error), hasta dar con la combinación que permita cumplir los requisitos funcionales. Este prototipo suele programarse haciendo uso de herramientas de muy alto nivel (p.e. Matlab) que facilitan el desarrollo rápido de aplicaciones pero que no son especialmente eficientes. En la mayoría de los casos, una vez validado este prototipo inicial se desecha, siendo necesario re-implementar cada algoritmo seleccionado, bien en su versión Sw (haciendo

uso de librerías Sw optimizadas), bien en su versión Hw (utilizando un lenguaje de descripción de Hw, p.e. VHDL[1]).

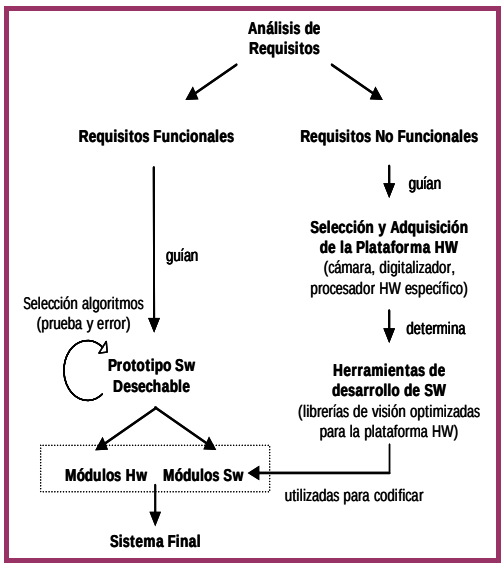


Figura 1. Esquema de las fases de desarrollo de un VIPS siguiendo el enfoque tradicional.

III. UNA NUEVA METODOLOGÍA DE DESARROLLO

Con el fin de solucionar los problemas derivados de la concepción Hw de los VIPS, se propone una nueva metodología que parte de los requisitos funcionales del sistema y permite retrasar la selección y adquisición de un determinado Hw hasta la fase final de implementación. La propuesta metodológica que aquí se presenta persigue el desarrollo de sistemas más flexibles, cuyos diseños sean reutilizables y fácilmente adaptables a distintas plataformas. Esta flexibilidad sólo es posible si se afronta la construcción de estos sistemas mixtos poniendo el énfasis en su componente Sw.

Como se describe a continuación, la construcción de VIPS suele requerir explorar una gran cantidad de configuraciones: distintas combinaciones de algoritmos de procesamiento de imágenes, diferentes particiones Sw/Hw de la funcionalidad y diversas plataformas sobre las que realizar la co-simulación e implementación final del sistema. Así, muy probablemente, durante el desarrollo de estos sistemas deberemos volver atrás en el diseño para reconsiderar alguna de las decisiones adoptadas. Por ello, la metodología que proponemos no consiste en una secuencia lineal de tareas sino en un proceso en espiral que permite refinar el sistema de manera iterativa e incremental hasta llegar a la versión final.

A pesar de que esta propuesta parte de algunos conceptos relativamente abstractos, propios de una metodología de desarrollo de Sw, fundamentalmente se pretende ofrecer una solución pragmática que dé soporte a todo el ciclo de vida de este tipo de productos, cubriendo las lagunas que tradicionalmente aparecen entre el diseño (abstracto) y la implementación (concreta).

A continuación se describe esta nueva metodología indicando, para cada una de las etapas identificadas (ver Fig. 2), qué tipo de herramientas pueden resultar útiles y si éstas existen o no actualmente en el mercado.

A. Prototipado funcional y simulación

La construcción y evaluación de prototipos es una forma rápida y económica de validar los requisitos de un sistema [12]. En concreto, durante esta primera fase se persigue validar los requisitos funciones del VIPS, creando un prototipo ejecutable que permita seleccionar los algoritmos de procesamiento de imágenes más adecuados en cuanto a su fiabilidad y robustez.

El prototipado funcional que se propone, a diferencia del realizado siguiendo el enfoque tradicional, es evolutivo y totalmente independiente de la plataforma sobre la que finalmente se implementen los algoritmos seleccionados.

Así, dado que no hay restricciones relativas al Hw empleado, este primer prototipo podrá implementarse haciendo uso de cualquiera de las librerías de procesamiento de imágenes disponibles en el mercado [6-8], incluso de aquellas optimizadas para una determinada plataforma, sea o no la utilizada finalmente. Dado que la eficiencia no es un factor determinante en este punto, podrá emplearse cualquier lenguaje de programación (C/C++, Matlab, Java, etc), siendo preferible uno de alto nivel que permita construir el prototipo lo más rápidamente posible y modificarlo de manera sencilla, si fuera necesario.

B. Particionado Sw/Hw y Co-simulación

Esta fase consiste en realizar una partición Sw/Hw de los algoritmos seleccionados durante el prototipado funcional, identificando cuáles de ellos se van a implementar como funciones Sw y cuáles como bloques Hw. El resultado de esta partición será un prototipo mixto Sw/Hw (co-prototipo) que se validará haciendo uso de técnicas de co-simulación.

A priori cualquier partición es viable, si bien debe procurarse reducir al máximo el intercambio de información entre elementos Sw y Hw para evitar retardos innecesarios y problemas de sincronización. Por lo general, resulta más útil realizar primero todo el procesamiento Hw (algoritmos sencillos de bajo nivel que acondicionen la imagen justo después de su adquisición), para posteriormente realizar el procesamiento Sw (algoritmos de procesamiento de alto nivel por lo general más complejos); de este modo, sólo será necesario realizar un único intercambio de datos Hw-Sw.

Una vez decidida esta partición es necesario seleccionar una plataforma que proporcione los bloques Hw necesarios, o al menos la funcionalidad para poder construirlos. Nótese que esta elección de una plataforma no implica necesariamente su adquisición ya que, en muchos casos, los fabricantes de estos dispositivos proporcionan modelos Sw lógicamente equivalentes sobre los que es posible simular el comportamiento del Hw de manera virtual. Este es el caso, por ejemplo, de la herramienta System Generator [13] que ofrece modelos Sw Simulink© [10] de las FPGA de Xillinx.

En cuanto al particionado, existen varias herramientas que permiten realizar este proceso de manera (semi-) automática [14]. Sin embargo, éstas suelen requerir especificar el prototipo inicial haciendo uso de la misma herramienta. Obviamente esto contraviene la filosofía de nuestra propuesta con la que se intenta promover la construcción de diseños independientes tanto de las herramientas de desarrollo como del Hw específico.

El mercado también ofrece varias herramientas de co-simulación con las que validar la viabilidad del co-prototipo (correcta sincronización Hw/Sw, tiempos de ejecución, ocu-

[1] VHDL: siglas de VHSIC-HDL (Very high speed integrated circuit Hardware Description Language).

pación de la memoria y el procesador, etc). Algunas, como la ya mencionada Simulink© [10], permiten realizar la co-simulación de manera tanto virtual (simulación Sw de la plataforma Hw) como real (los bloques Hw se ejecutan de manera efectiva sobre la plataforma seleccionada y los resultados se envían a Simulink© en tiempo real).

C. Implementación del código final

Una vez validado el co-prototipo sólo resta generar el código final de la aplicación y volcar cada parte sobre la plataforma Sw o Hw correspondiente. Por suerte, la mayoría de las herramientas de co-simulación permiten realizar este paso de manera (semi-) automática.

IV. LA HERRAMIENTA IP-CODER

Entre las herramientas de procesamiento de imágenes actualmente disponibles en el mercado podemos encontrar librerías de funciones Sw [6-8] y Hw [13] así como entornos completos de programación [9-10]; productos COTS [7-8] (comerciales, estándar y cerrados) y librerías de código abierto y libre distribución [6]; soluciones optimizadas para un determinado procesador de propósito general (p.e. OpenCv [6] optimizada para las MMX de Intel©) y otras diseñadas para una determinada plataforma Hw (p.e. los IP-cores de Xillinx© [13] optimizados para sus FPGAs).

Dada la enorme cantidad y variedad de estos productos cabe preguntarse por qué ninguno de ellos cubre todas las fases implicadas en el proceso de construcción de los VIPS. Muy probablemente la respuesta a esta pregunta se halle en la ausencia de estándares que faciliten la integración de los distintos productos. De hecho, estas herramientas están implementadas utilizando distintos lenguajes y paradigmas de programación y suelen definir sus propios tipos de datos y convenciones de paso de parámetros para sus funciones, esto sin mencionar la concepción radicalmente distinta de los productos orientados al Sw (procesamiento secuencial del código) frente a los más orientados al Hw (procesamiento paralelo de los datos).

Así, como afirma Perrier en su artículo [15], es necesario desarrollar una nueva generación de aplicaciones que, de una parte, resuelva los vacíos existentes entre las herramientas actuales, y de otra, permita la integración de Sw y Hw de una forma más natural. En esta línea, e intentando evitar "reinventar la rueda" cuando sea posible, IP-CoDER se construye más como un entorno integrador de distintas herramientas que como un nuevo producto para el desarrollo de VIPS. A continuación se describe cómo, gracias al uso de técnicas de generación automática de Sw y a la definición de un lenguaje de programación visual sencillo e intuitivo de manejar, IP-CoDER permite integrar una amplia variedad de herramientas de manera transparente para el usuario.

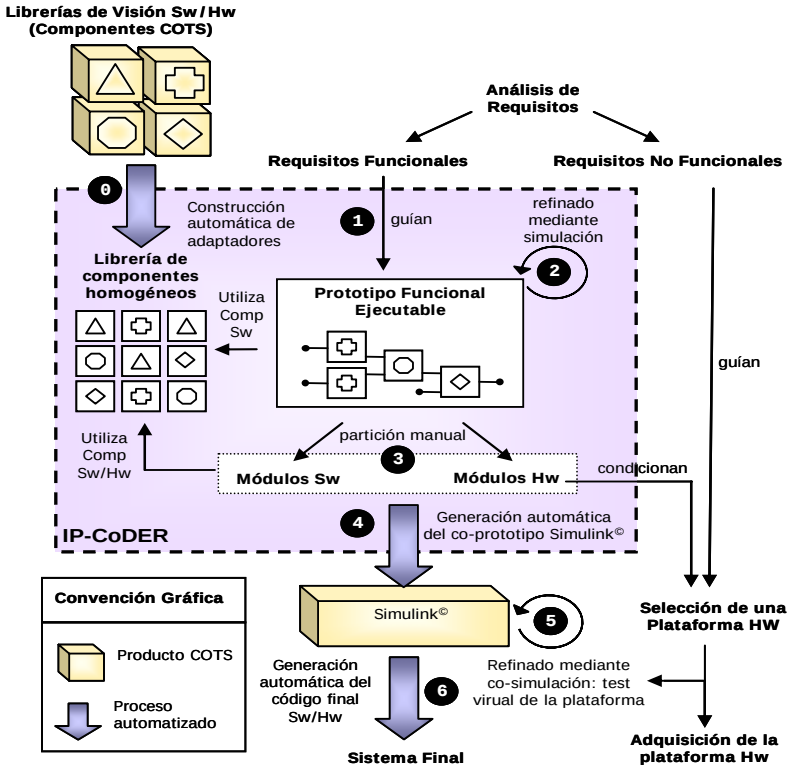


Figura 2. Esquema del ciclo de vida de desarrollo de VIPS haciendo uso de la herramienta IP-CoDER.

A. Biblioteca de componentes homogéneos sw/hw para procesamiento de imágenes

En este apartado se describe cómo se han confeccionado sendas librerías de componentes homogéneos Sw y Hw que encapsulan funciones procedentes de varias de las librerías de procesamiento de imágenes existentes (ver Fig. 2 - Etapa 0). Estos componentes exhiben interfaces compatibles de modo que el usuario puede conectarlos entre sí, reemplazar unos por otros, o agruparlos para construir componentes de más alto nivel, sin tener que preocuparse de la compatibilidad entre las funciones contenidas internamente en dichos componentes. El procedimiento de creación de nuevos componentes se ha diseñado de modo

que es posible incorporar las funciones de tantas librerías como se desee, incluso de aquellas que pudieran aparecer en el futuro. De este modo se facilita el mantenimiento y la extensibilidad tanto de las librerías de componentes como de las aplicaciones que hacen uso de ellas.

Tanto los componentes Sw como los Hw hacen uso adaptadores o *wrappers* para conseguir homogeneizar tanto sus interfaces como los tipos de datos que intercambian con otros componentes [16]. En ambos casos, se han modelado tanto componentes simples (contienen funciones de librería) como componentes compuestos (contienen dos o más componentes previamente definidos ya sean simples o compuestos).

Los adaptadores de los componentes Sw se generan utilizando la herramienta IP-CoDER. Para ello el usuario deberá rellenar una plantilla especificando, entre otras cosas, los conectores del componente (interfaz externa), los parámetros de la función que desea encapsular (interfaz interna), la correspondencia entre ambas interfaces, y en el caso de componentes simples, la librería de la que procede la función interna. A partir de estos datos IP-CoDER generará automáticamente el código asociado al nuevo componente (Fig. 3.a y 3.b), realizando las transformaciones de tipos necesarias tanto de los parámetros de entrada como de los de salida, e invocando a la función original en el formato adecuado. Nótese que en el caso de los componentes compuestos no es necesario realizar ninguna transformación de tipos (Fig. 3b) puesto que los componentes internos ya son homogéneos.

En el caso de los componentes Hw, si bien la herramienta utilizada para construir los *wrappers* es Simulink©, el procedimiento es idéntico (relleno de una plantilla genérica). La elección de Simulink© en lugar de IP-CoDER para esta tarea responde fundamentalmente a dos razones. En primer lugar, como ya hemos comentado, varias de las librerías Hw existentes proporcionan modelos Sw Simulink© equivalentes [13], por lo que en estos casos no hará falta realizar adaptaciones adicionales. Además, los componentes Hw intervienen fundamentalmente durante la co-simulación, y esta se realiza utilizando Simulink©, tal y como se explica en el apartado 4.3.

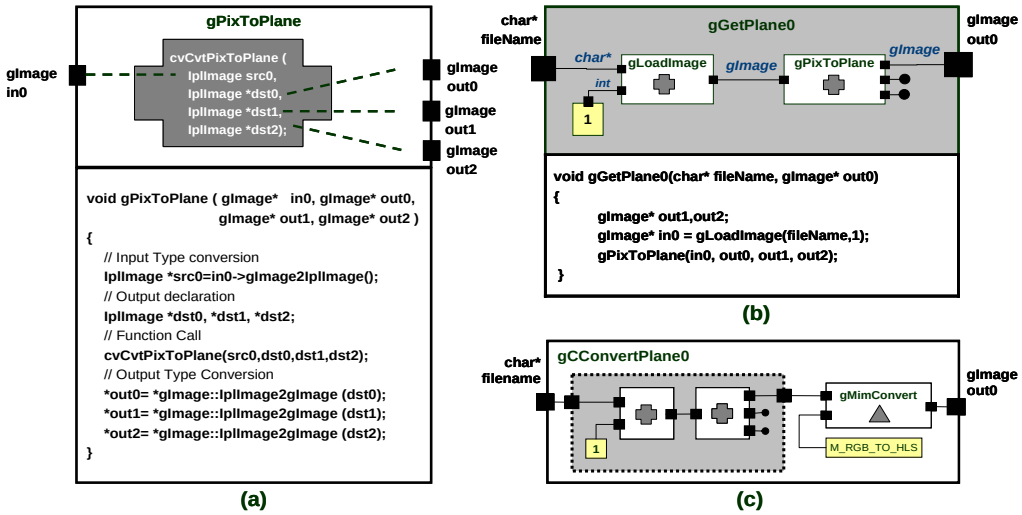


Figura 3. Componentes Sw para procesamiento de imágenes. **(a)** Componente simple generado automáticamente a partir de una función OpenCV [3], **(b)** y **(c)** Componentes compuestos a partir de los definidos previamente.

B. Prototipado funcional con ip-coder

IP-CoDER cubre la fase de prototipado funcional definida en la metodología (ver Fig. 2 - Etapa 1) durante la que se seleccionan los algoritmos de procesamiento de imágenes adecuados para cada VIPS. Dado que IP-CoDER se concibió como una herramienta de programación visual, se ha dotado a los componentes Sw de la biblioteca de una interfaz gráfica de modo que el usuario puede seleccionarlos, arrastrarlos, conectarlos entre sí y agruparlos con un simple clic del ratón. Cuando el prototipo está listo (ver Fig. 4.a) IP-CoDER genera automáticamente el código ejecutable equivalente, enlazando el código de sus componentes atendiendo a cómo éstos están conectados. Este código permite simular el funcionamiento del VIPS utilizando distintas imágenes de entrada. Si el resultado no es el esperado el usuario podrá añadir, eliminar, o cambiar los componentes hasta encontrar una configuración más adecuada.

Así, hemos pasado del enfoque tradicional en el que este prototipo se codificaba manualmente haciendo uso de una única librería (la optimizada para la plataforma previamente seleccionada), a poder construir un prototipo funcional sin escribir ni una sola línea de código y en el que podemos hacer uso de varias librerías simultáneamente gracias a la capacidad integradora de IP-CoDER.

C. Generación del co-prototipo Simulink©

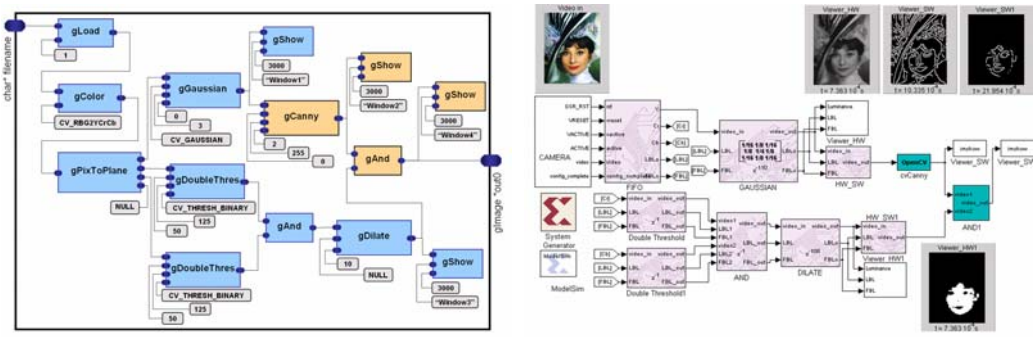
Una vez validado el prototipo funcional, el usuario podrá realizar una partición manual Sw/Hw del mismo (Fig. 2 - Etapa 3) previa elección de una plataforma Hw. IP-CoDER le ofrecerá los componentes Hw disponibles en la biblioteca para la
plataforma seleccionada, de modo que pueda elegir cuáles utilizar en la co-simulación. Una vez particionado el prototipo, IP-CoDER generará automáticamente el co-prototipo Simulink© correspondiente (Fig. 2 - Etapa 4), creando los adaptadores Simulink© correspondientes para los bloques Sw (*S-function*) e invocando a los bloques Hw seleccionados en la librería.

D. Generación del código final del sistema

Una vez validado el co-prototipo, el código final Sw y Hw se genera automáticamente por separado. Así, mientras Simulink© genera el código VHDL correspondiente a los bloques Hw, IP-CoDER genera el código correspondiente a la partición Sw del prototipo inicial (Fig. 2 - Etapa 6).

E. Ejemplo de uso

Para validar la utilidad y el correcto funcionamiento de IP-CoDER se ha desarrollado un VIPS para la detección de zonas de piel en imágenes color cuyos resultados se muestran en la Fig. 4. Es necesario reseñar que a pesar de los cambios introducidos durante el prototipado funcional, su implementación llevó tan sólo unos minutos. Por otra parte, y a pesar de lo pequeño de las imágenes empleadas, la co-simulación tardó en ejecutarse aproximadamente 15 minutos. De haberse empleado imágenes más grandes este tiempo podría ascender a horas. Esto nos lleva a la conclusión de que, como cabía esperar, los cambios introducidos durante el prototipado funcional tienen un impacto mucho menor en el tiempo de desarrollo de las aplicaciones que los realizados durante el co-prototipado.



(a) **(b)**

Figura 4. **(a)** Prototipo funcional IP-CoDER en el que se ha realizado una partición Hw (azul) / Sw (naranja), **(b)** Co-Prototipo Simulink® obtenido a partir de (a); encima se muestran las imágenes resultado de la co-simulación.

IV. CONCLUSIONES Y TRABAJOS FUTUROS

Este trabajo presenta una nueva metodología de desarrollo de sistemas mixtos Sw/Hw que pretende la obtención de diseños flexibles, reutilizables y fácilmente adaptables a distintas plataformas. La metodología propuesta aborda todas las fases del ciclo de vida de estos sistemas siguiendo un proceso en espiral. Si bien esta propuesta se centra en el dominio de los VIPS, los resultados obtenidos son fácilmente trasladables a otros sistemas como los de control o los de telecomunicación.

Para dar soporte a la metodología propuesta se ha desarrollado la herramienta visual IP-CoDER que permite integrar distintas librerías, en su mayoría COTS, de modo que puedan utilizarse conjuntamente. Esta herramienta permite además la generación automática de prototipos funcionales ejecutables y de co-prototipos Hw/Sw de VIPS.

En el futuro, sería interesante conseguir un mayor grado de automatización del proceso de particionado Sw/Hw, así como contar con un sistema de ayuda a la decisión que guíe al usuario en la selección de los algoritmos de procesamiento de imágenes más adecuados en cada caso.

V. AGRADECIMIENTOS

Queremos agradecer a A. J. Martínez Lamberto y a J. A. Martínez Navarro su colaboración en el desarrollo de IP-CoDER.

VI. REFERENCIAS

[1] C. Vicente, C. Fernández, P. Sánchez, "*Desarrollo de Sistemas de Inspección Visual Automatizada a partir de la Descripción de un Patrón Arquitectural Genérico*", NOVATICA, no. 171, pp. 63-65, 2004.

[2] C. Vicente, A. Toledo, C. Fernández, "*Heterogeneous COTS product integration to allow the comprehensive development of image processing systems*", 4th Int'l Conf. on COTS-Based Software Systems, LNCS Vol. 3412, pp. 8, Bilbao, 2005.

[3] K. Czarnecki and U. Eisenecker, *Generative programming: methods, tools, and applications*, Addison-Wesley, 2000.

[4] F. Bachmann et al.; "*Technical concepts of component-based software engineering*", Tech. report CMU/SEI-2000-TR-0008, Software Engineering Institute (SEI), Carnegie Mellon Univ., 2000.

[5] C. Albert, L. Brownsword, "*Evolutionary process for integrating COTS-based systems (EPIC): An Overview*", Tech. report CMU/SEI-2002-TR-009, Software Engineering Institute (SEI), Carnegie Mellon Univ., 2002.

[6] Intel® Open Source Computer Vision Library, información disponible en: *www.intel.com/research/mrl/research/opencv*

[7] Matrox Imaging Library (MIL) v.7.5, información del producto disponible en: *www.matrox.com/imaging/products/mil*

[8] Matlab® Image Processing Toolbox v.5, información del producto disponible en: *www.mathworks.com/products/image/*

[9] VisiQuest, información disponible en: *http://www.accusoft.com/imaging/visiquest*

[10] Simulink® v.6, información disponible en: *www.mathworks.com/products/simulink/*

[11] SIMNON for Windows. información disponible en: *www.mpassociates.gr/software/ catalog/sci/simnon/simnon.html*

[12] I. Sommerville, *Software Engineering 6th Edition*, Chap. 8, Addison-Wesley, 2000.

[13] Xillinx System Generator, información del producto disponible en: *www.xilinx.com*

[14] GEDAE, información del producto disponible en: *www.gedae.com*

[15] V. Perrier, *"A look inside Electronic System Level (ESL) design"*, EEDesign.com, Art. Id. 18402916, 2004.

[16] J.M. Troya, and A. Vallecillo, "*Controllers: reusable wrappers to adapt software components*", Information & Software Technology, Vol. 43, No. 3, pp. 189-202, 2001.

VII. BIOGRAPHIES

Cristina Vicente-Chicote is an Assistant Professor and a PhD candidate in Telecommunication Engineering at the Tech. Univ. of Cartagena (Spain). Her research interests include software product lines, COTS product integration, generative and visual programming, and image processing applications. She received a MS in Computer Science from the Univ. of Murcia (Spain). She is a member of the Systems and Electronic Engineering Division (SEED) research group of the Tech. Univ. of Cartagena (Spain). Contact her at Cristina.Vicente@upct.es.

Ana Toledo-Moreo is a PhD and Assistant Lecturer in the Electronics Technology Department of the Technical University of Cartagena, Spain. Her research has been mainly focussed in developing low-level image processing algorithms for FPGAs. She is also interested in artificial neural networks, principally in the improvement of learning algorithms. She has a number of publications related with the two areas in international journals and conferences. Contact her at Ana.Toledo@upct.es.

Carlos Fernández-Andrés is an Associate Professor of Computer Science at the Tech. Univ. of Cartagena (Spain). His research interests include conceptual image processing applications, software and hardware architectures, and software/hardware co-design. He received a PhD in Industrial Engineering from the Tech. Univ. of Madrid (Spain). He is a member of the Systems and Electronic Engineering Division (SEED) research group of the Tech. Univ. of Cartagena (Spain). Contact him at Carlos.Fernandez@upct.es.

Pedro Sánchez-Palma is an Associate Professor at the Tech. Univ. of Cartagena (Spain). His research interests include conceptual modelling, software architectures and object-oriented paradigm. He received a PhD in Computer Science from the Tech. Univ. of Valencia (Spain) in 2000. He is a member of the Systems and Electronic Engineering Division (SEED) research group of the Tech. Univ. of Cartagena (Spain). Contact him at Pedro.Sanchez@upct.es.